**Web前端工程化方案**

**目 录**

[第一章 前端工程化 3](#_Toc444002417)

[第一节 什么是前端工程化 3](#_Toc444002418)

[第二节 为什么需要前端工程化 3](#_Toc444002419)

[第二章 自动化内容 3](#_Toc444002420)

[第一节 文件合并压缩 3](#_Toc444002421)

[第二节 Html拼接 4](#_Toc444002422)

[第三节 sass编译 4](#_Toc444002423)

[第四节 css雪碧图 4](#_Toc444002424)

[第五节 图片无损压缩 4](#_Toc444002425)

[第六节 缓存更新 4](#_Toc444002426)

[第七节 js代码质量检测 4](#_Toc444002427)

[第八节 js单元测试 4](#_Toc444002428)

[第九节 端对端测试 4](#_Toc444002429)

[第三章 前端工程化工具及使用 4](#_Toc444002430)

[第一节 环境 4](#_Toc444002431)

[第二节 工具 4](#_Toc444002432)

[第三节 目录结构 5](#_Toc444002433)

# 前端工程化

## 工程化

借用苏宁前端架构师(@xufei)的一个总结来说明工程化。

编程技术及生态发展的三个阶段：

* 最初的时候人们忙着补全各种API，代表着他们拥有的东西还很匮乏，需要在语言跟基础设施上继续完善。
* 然后就开始各种模式，标志他们做的东西逐渐变大变复杂，需要更好的组织了。
* 然后就是各类分层MVC，MVP，MVVM之类，可视化开发，自动化测试，团队协同系统等等，说明重视生产效率了，也就是所谓工程化。

## web前端的发展

随着时间的推移，web前端从一开始的切个信息展示的页面，到复杂的pc web app，再到移动端app。前端开发面向的已经不是页面，而是应用了。前端的开发模式也更接近于传统GUI软件，而且由于现在前端领域为了解决日益复杂的web业务需求及体量，越来越多的借鉴了传统客户端的开发经验，导致两者变得越来越趋同。再加上前端一些独特的特性(免安装、增量安装等)，工程上的复杂度有过之而无不及。

## web前端工程化

1. **前端工程化的来源**

前端三种完全不一样的语言，加上跨浏览器、跨平台，而且前端一直没有统一的规范。导致不同公司，甚至相同公司不同项目的前端规范完全迥异。所以前端无法像java有IDE或者类似Maven的工具用于规范管理代码，前端团队只能利用其它语言编写的工具来分别处理合并，压缩等问题。

Nodejs的发布，使前端工程师脱离了浏览器的束缚。大量前端工具出现在前端生态中，诸如模块规范（CMD、AMD），构建工具（Grunt、Gulp），第三方库管理（nmp、bower）等。前端终于脱离茹毛饮血的时代，进入了工业化时代，利用工程化工具自动化处理很多繁琐的工作。

1. **前端工程化解决问题**
2. 合理的开发流程及开发规范，包括代码规范、模块化组件化规范等（提高生产力）。
3. 一套自动化代码质量检测方案（提高系统可靠性）。
4. 一套自动化及高度适应性的项目 发布／部署 方案（提高系统的伸缩性及灵活性）。
5. 极致的性能优化，包括减少冗余的接口请求及资源请求、提高缓存命中率等，简言之就是站点的打开及运行速度（更好的用户体验）。

# 前端工程化工具及使用

## 环境

* Node.js：目前前端工程化工具都是基于node.js开发。并且前端能利用node.js快速起页面渲染服务器，利用一些Mock工具脱离后端，进行独立开发。
* Ruby：用于编译sass。

## 工具

1. NPM

NPM的全称是Node Package Manager，是一个node.js包管理和分发工具，它的主要功能就是管理node包，包括：安装、卸载、更新、查看、搜索、发布等。在服务器端的node.js中，NPM用于管理第三方包。而web前端只利用NPM管理开发过程用到的工具。

1. **Bower**

Bower是一个客户端技术的软件包管理器，它可用于搜索、安装和卸载如JavaScript、HTML、CSS之类的网络资源。简单说就是前端的maven。

1. **Gulp**

Gulp 是一款基于任务的设计模式的自动化工具，通过插件的配合解决全套前端解决方案，如静态页面压缩、图片压缩、JS合并、SASS同步编译并压缩CSS、服务器控制客户端同步刷新。

1. **Karma**

Karma是一个基于Node.js的JavaScript测试执行过程管理工具（Test Runner）。该工具可用于测试所有主流Web浏览器，也可集成到CI（Continuous integration）工具，也可和其他代码编辑器一起使用。Karma的一个强大特性就是，它可以监控(Watch)文件的变化，然后自行执行，通过console.log显示测试结果。

1. **Jasmine**

Jasmine 是一款 JavaScript 测试框架，它不依赖于其他任何 JavaScript 组件。它有干净清晰的语法，简单的测试代码。

1. **Protractor**

Protractor 是 AngularJS 团队构建的一个端对端的测试运行工具，模拟用户交互，验证web应用的运行状况。”

Protractor使用Jasmine测试框架来定义测试。Protractor为不同的页面交互提供一套健壮的API。

# 自动化内容

## 文件合并压缩

1. **文件合并**

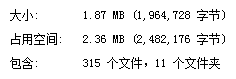
js和css文件合并能有效减少资源请求数量。并且前端当采用了模块和组件开发后，将会产生大量js文件，文件合并就必不可少。

1. **文件压缩**

文件压缩将去除文件中的注释、空格和回车，并且对js中变量替换为更短的字母组合，可以大大减少文件大小，降低网络资源消耗。

1. **效率**

压缩效率主要取决于注释数量和变量名长度。 采用某框架的js文件做测试。

合并压缩前：

合并压缩后：C:\Users\Administrator\Desktop\QQ截图20160225171433.png

压缩率为33.5%。

## Html拼接

对html文件进行预处理，将头尾html文件，在发布前就拼接好，减少运行时再jsp拼接的资源消耗。

如果采用组件化开发，也需要该功能拼接声明的各个组件，生成对应的页面。

## sass编译

对sass文件进行编译，生成css文件。

## css雪碧

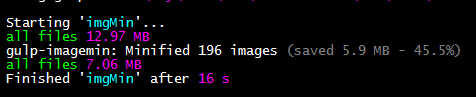
CSS雪碧 即CSS Sprite，是一种CSS图像合并技术，该方法是将小图标和背景图像合并到一张图片上，然后利用css的背景定位来显示需要显示的图片部分。

该功能能通过检查css文件，将其中声明需要sprite的图片合成为一张图片，并且自动修改css文件里的内容，加入background-position属性，正确显示原有图像。

Css sprite能减少资源请求数量。

## 图片无损压缩

通过Photoshop等软件切出的图片会带有很多无关信息，删除这些信息能减少图片大小。下图可见，压缩效率能达到55%。



## 缓存更新

为了提升浏览器打开页面的速度，一般都会让浏览器直接缓存图片或文件，不用每次向服务器请求资源。但版本更新后，浏览器还会使用缓存中旧文件，造成页面错误。

解决缓存更新问题，一般采用更改文件名或增加请求后缀。使用前端构建工具能根据文件内容计算md5，同时修改文件名和引用地址。这样非覆盖更新，还能解决CDN部署不同步的问题。

## js代码质量检测

## js单元测试

## 端对端测试

# 工具安装及使用

## 目录结构及文件说明